

# Advanced Fuzzing with Peach 2



MICHAEL EDDINGTON

[MIKE@LEVIATHANSECURITY.COM](mailto:MIKE@LEVIATHANSECURITY.COM)



# Agenda



- Introduction to Peach 2
- Data mutations
- Peach State Machine
- Peach Farm
- Peach in The Middle

# Introduction to Peach 2



# Peach 1



- Framework for writing fuzzers
- Instrumentation via wrapper APIs
- No data definition layer (DDL), just fuzzer
- Steep learning curve
- Complex fuzzers result in complex fuzzer code

# Peach 2



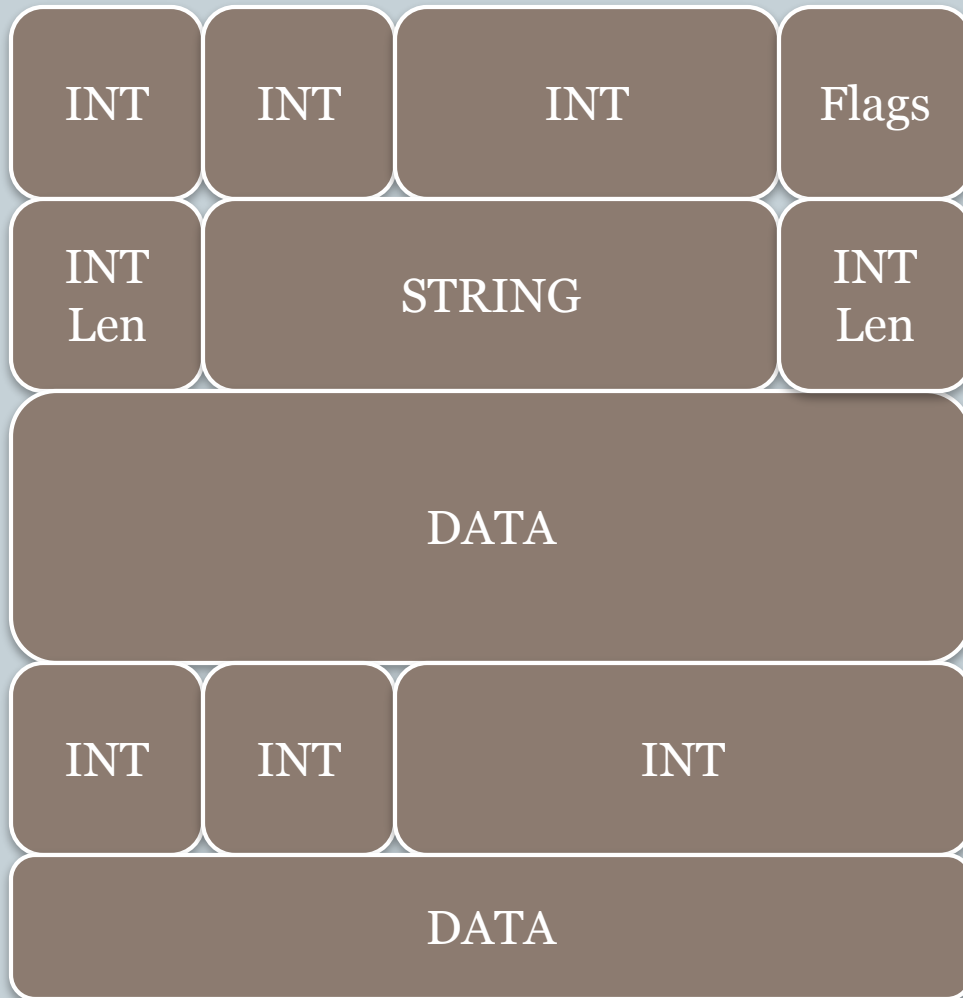
- Reduce creation time and simplify fuzzer generation
- Fuzzer platform, not framework
- Modeling based approach
- Fault detection
- Lower learning curve

# Modeling Based Fuzzing

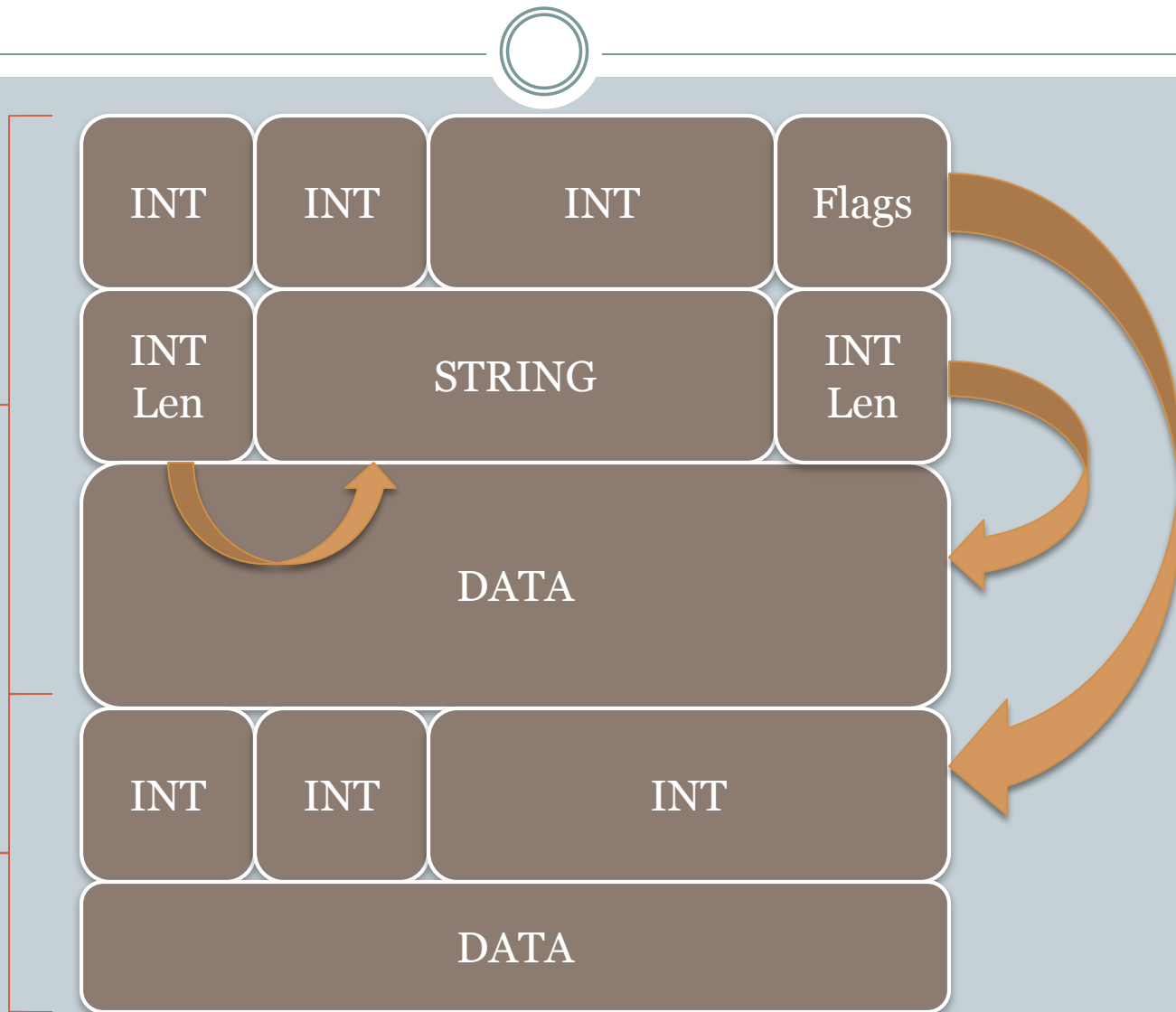


- Model types and data
- Model state machine
- Support models with data sets
- Mutate models with mutators

# Model Data: Types

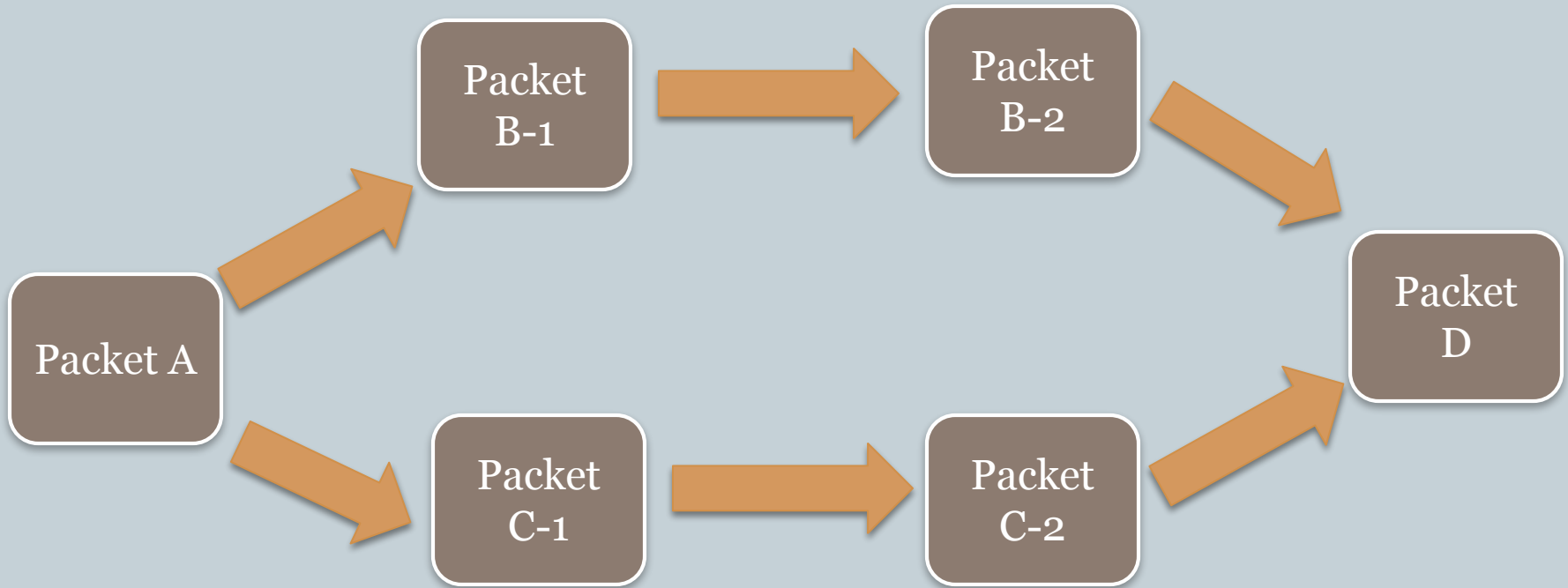


# Model Data: Relationships





# Model Data: State Model



# Benefits of Modeling



- Easy reuse of definitions
- Complex mutations can be applied to a model
- Improvements to data generation or mutation independent of model
- Data read into definition as well as generated

# Data Modeling

- Define structure of data
  - Define relations in data
  - Reuse definitions
- Block
  - Sequence
  - Choice
  - String
  - Number
  - Flags/Flag
  - Blob
- 
- Relation
  - Transformer

# State Modeling



# State Modeling



## Stream

- TCP, UDP, Files
- Connect
- Accept
- Input
- Output
- Close

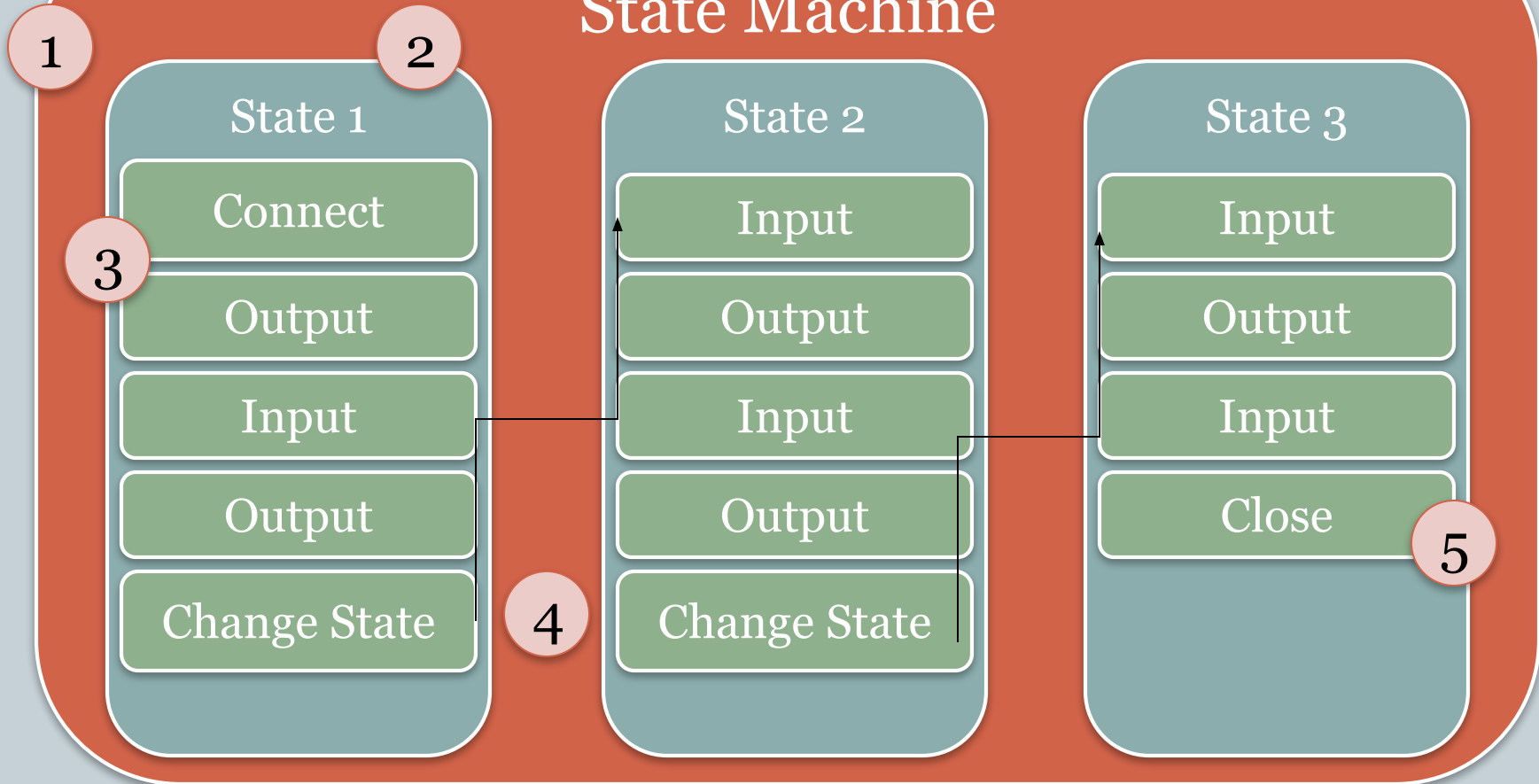
## Call

- COM, RPC, SOAP
- Call
  - Method
  - Parameters
  - Result

# State Modeling: Stream



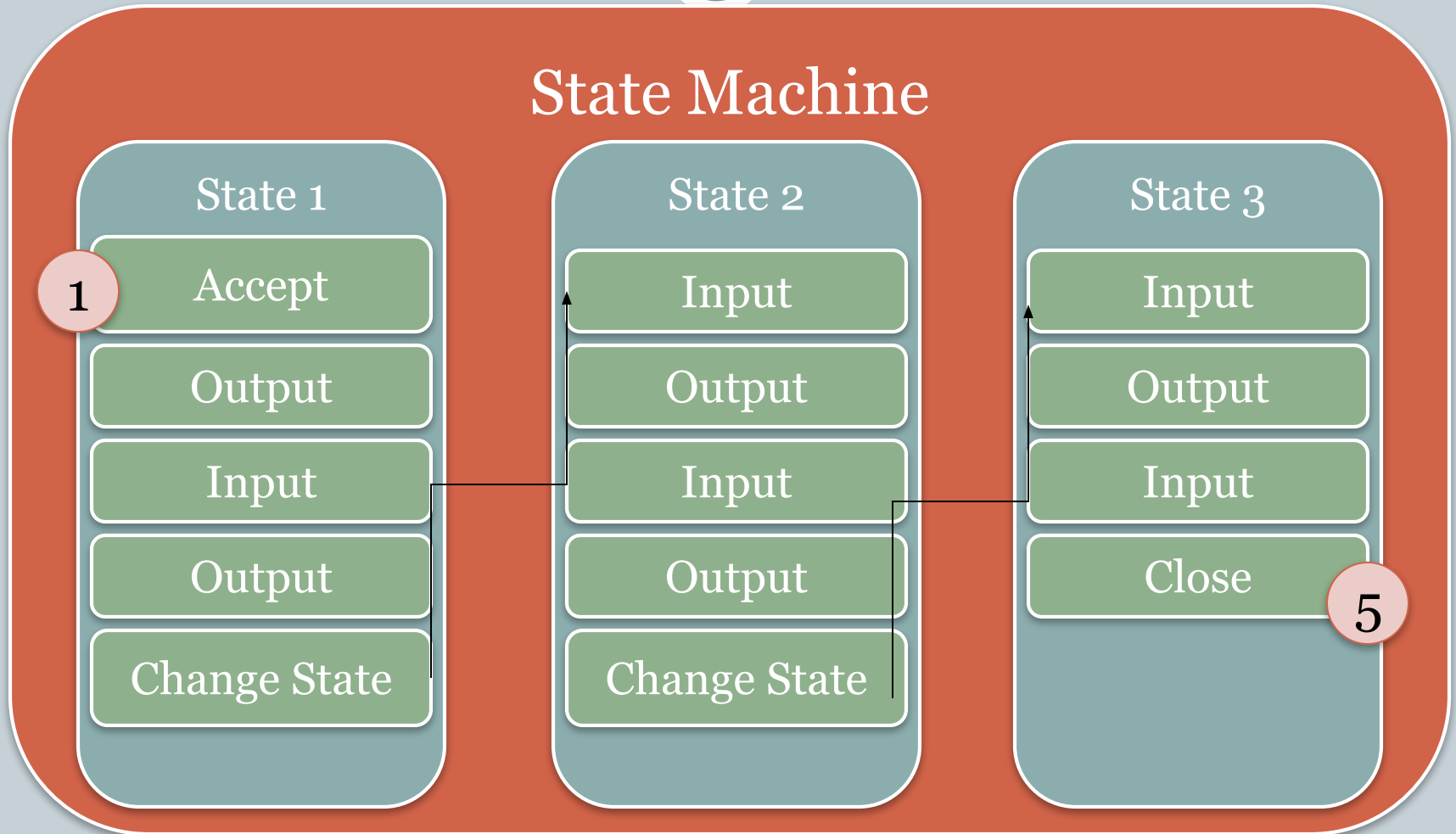
## State Machine



# State Modeling: Stream



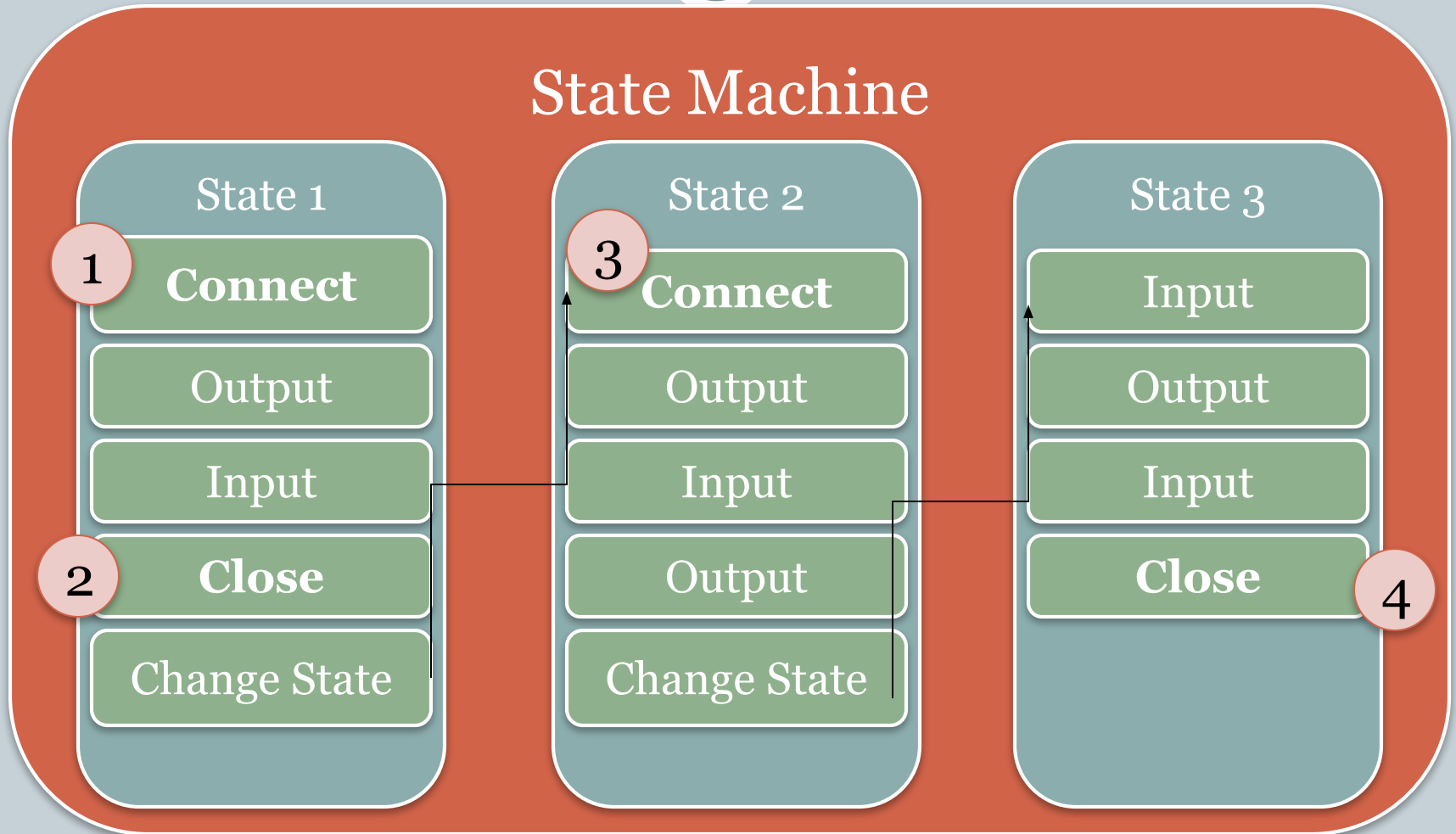
## State Machine



# State Modeling: Stream



## State Machine

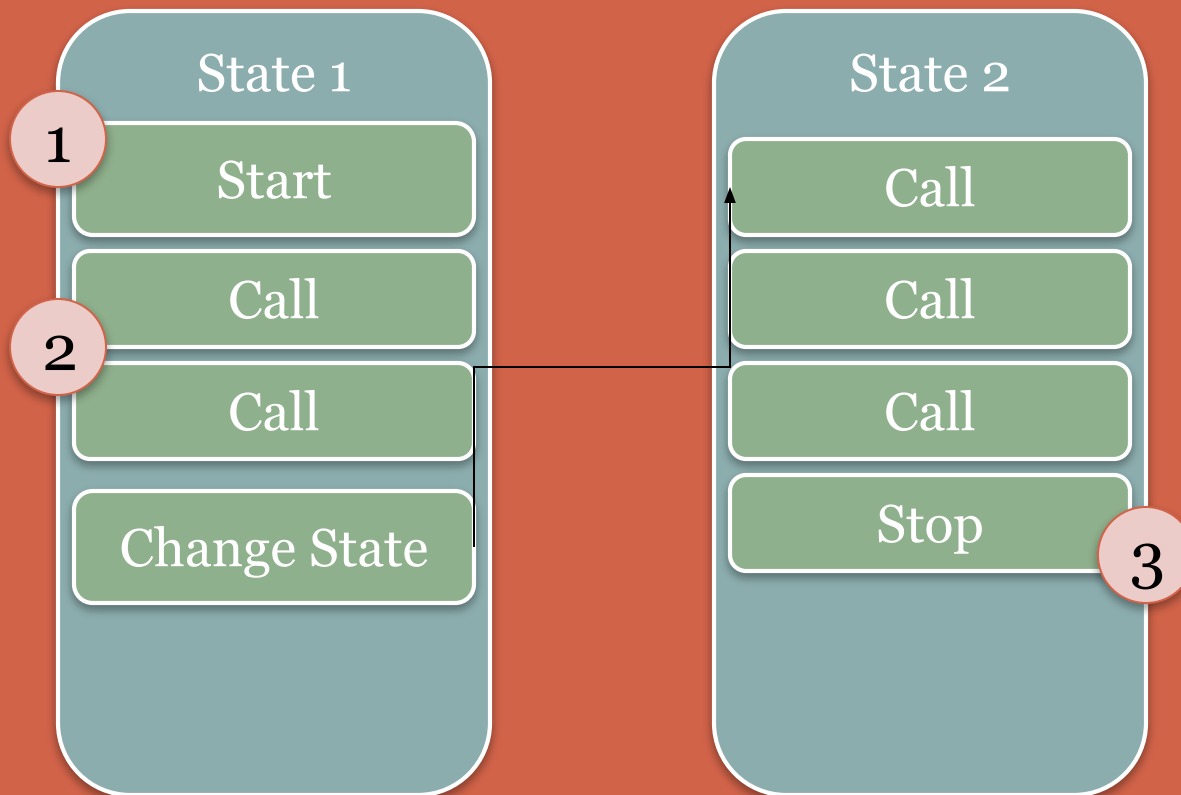




# State Modeling: Call



## State Machine



# Data Mutations

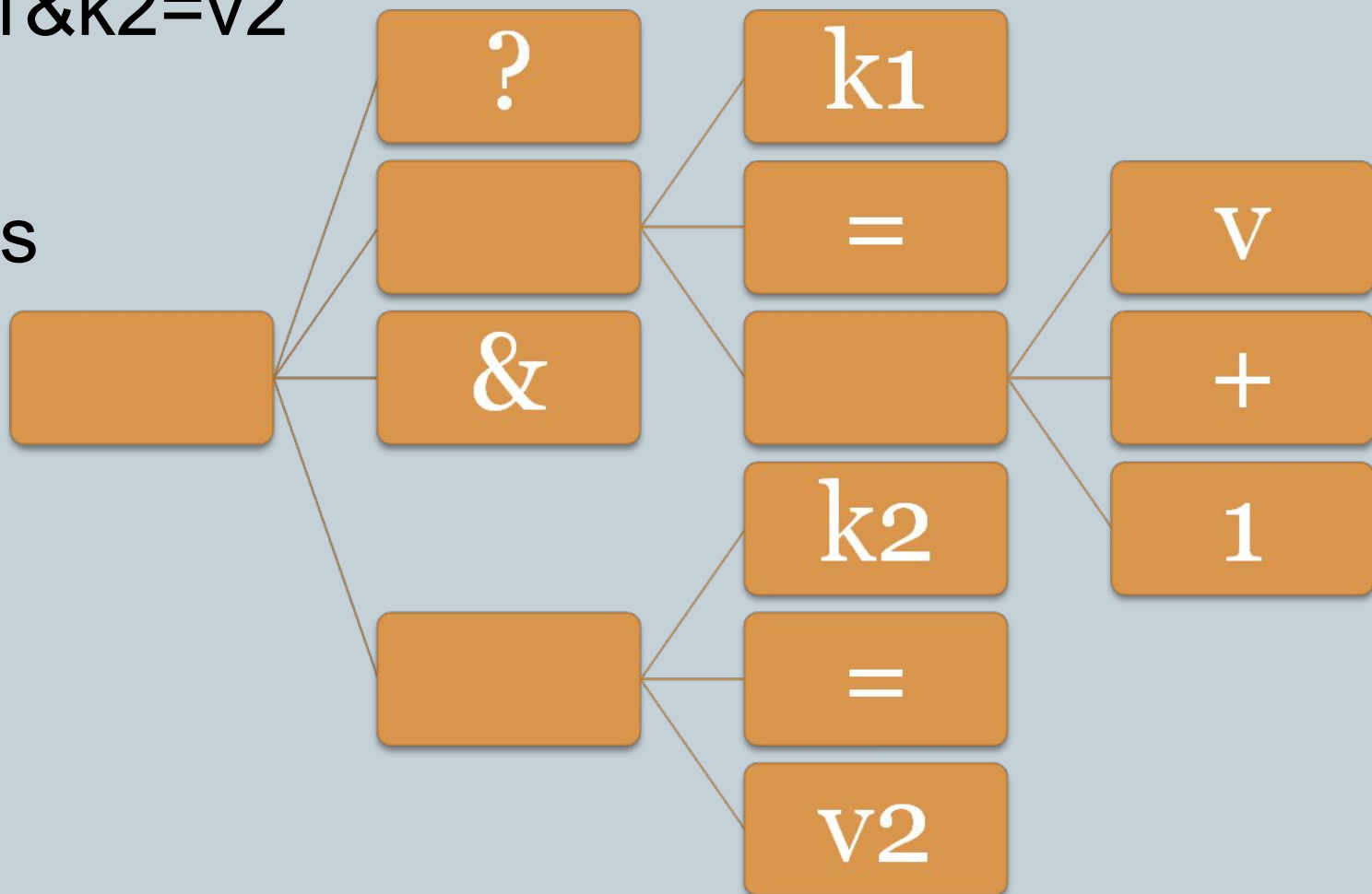


# Mutation: String



“?k1=v+1&k2=v2”

40,000+  
variations

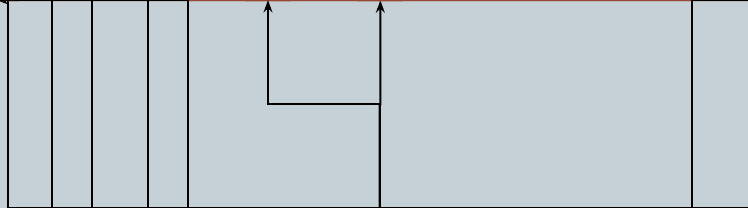
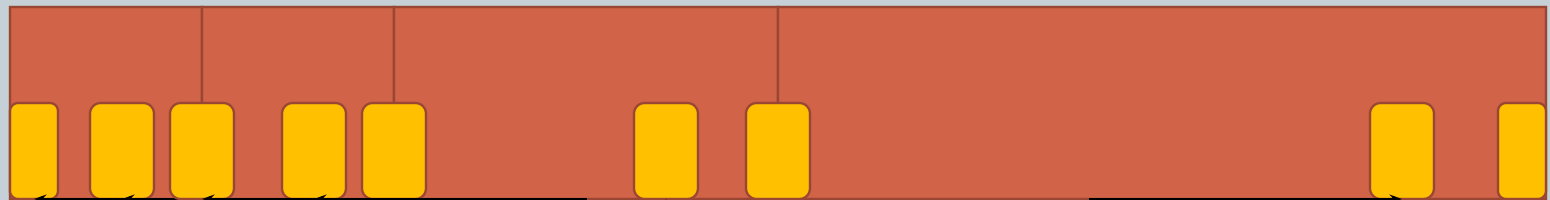


# Mutation: Number



00

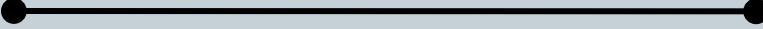
FFFFFFFFFFFFFFFFFFFF



Interesting Edge Cases

# Mutation: Size Relation #1



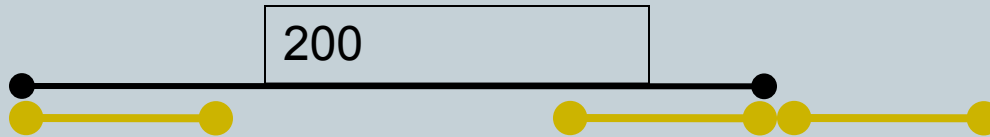
● Length:  200

● Data:  200 Bytes

# Mutation: Size Relation #2



● Length:



● Data:



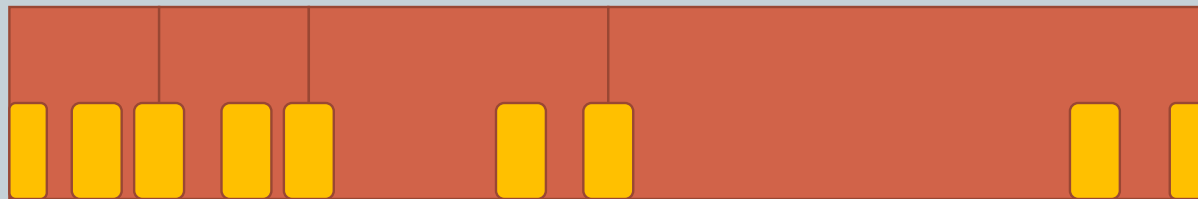
# Mutation: Size Relation #3



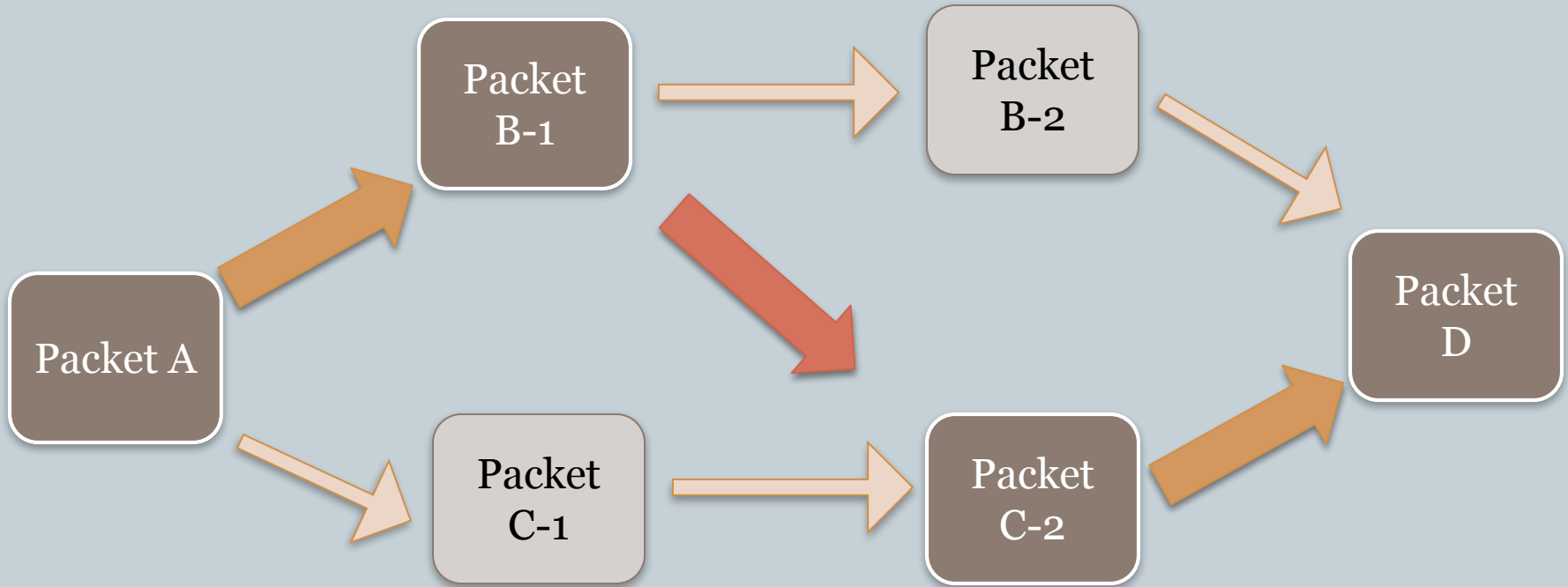
- Data & Length:

00

FFFFFFFFFFFFFFFF

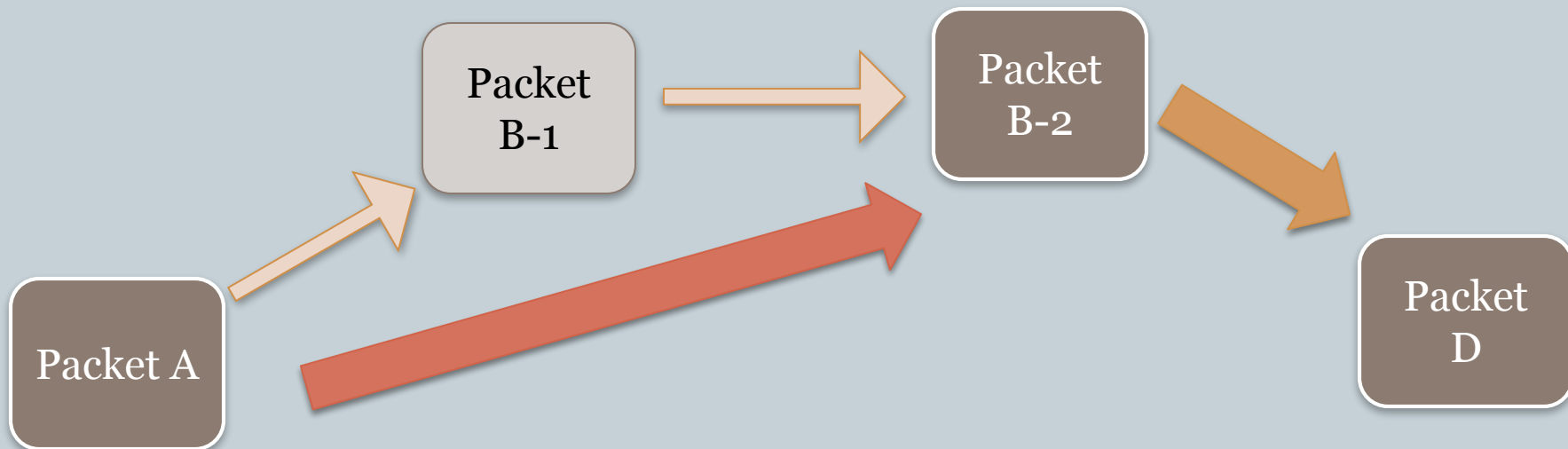


# Mutation: State

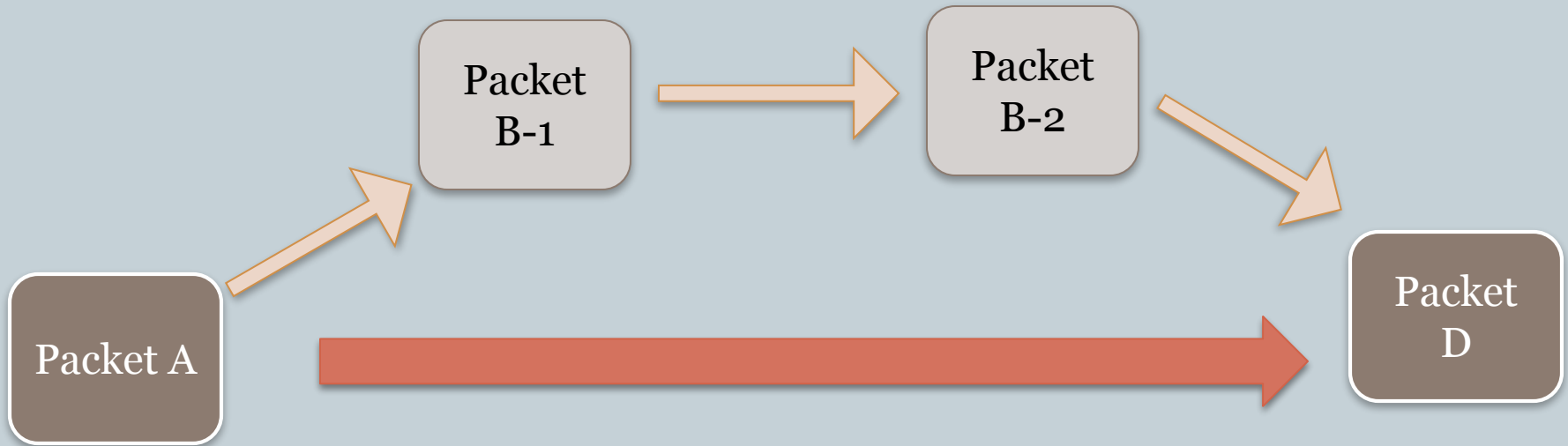




# Mutation: State



# Mutation: State



# Add Custom Mutators



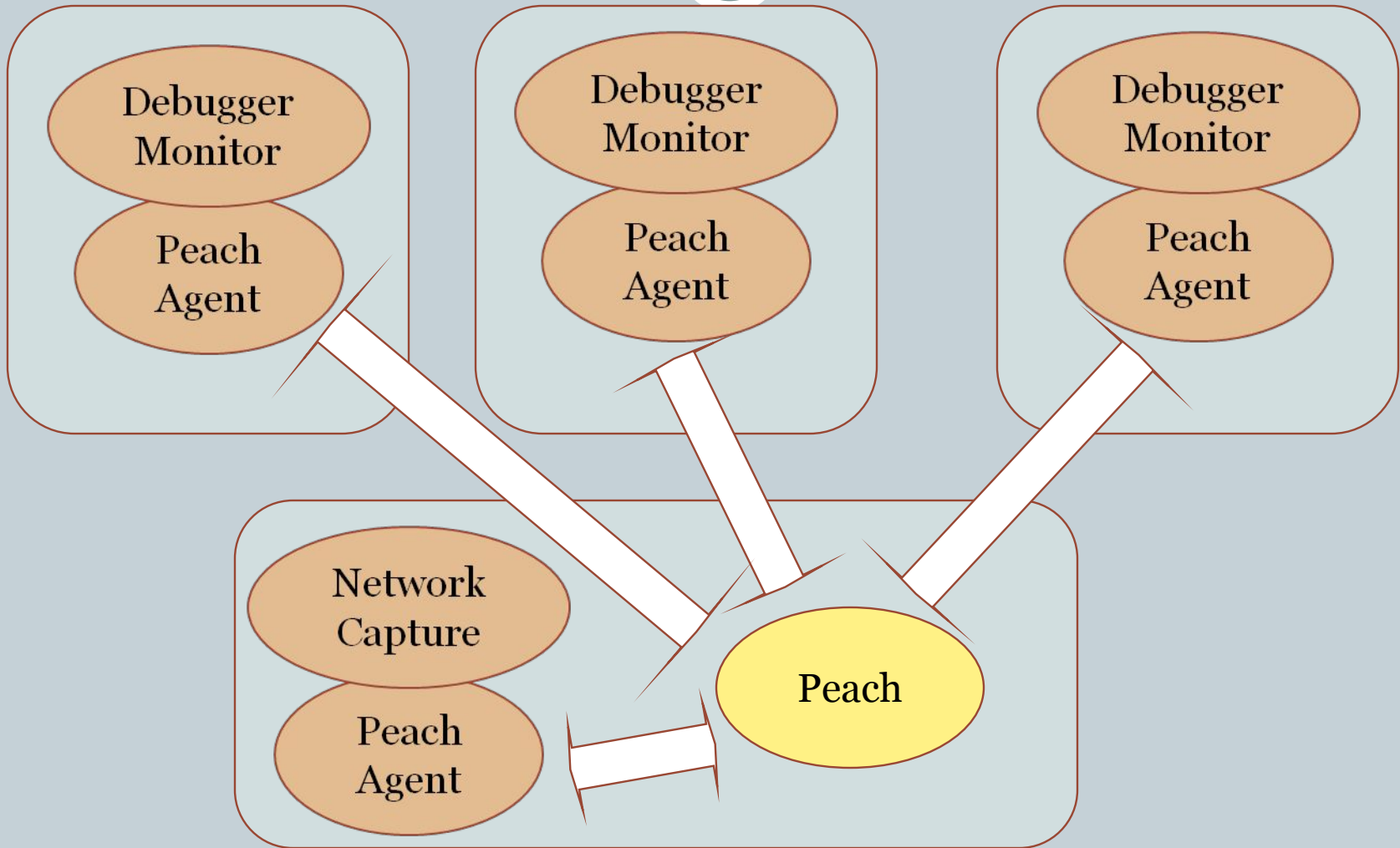
- Sling some Python
- Add additional mutations
- Specific mutations
- Etc.

# Fault Detection

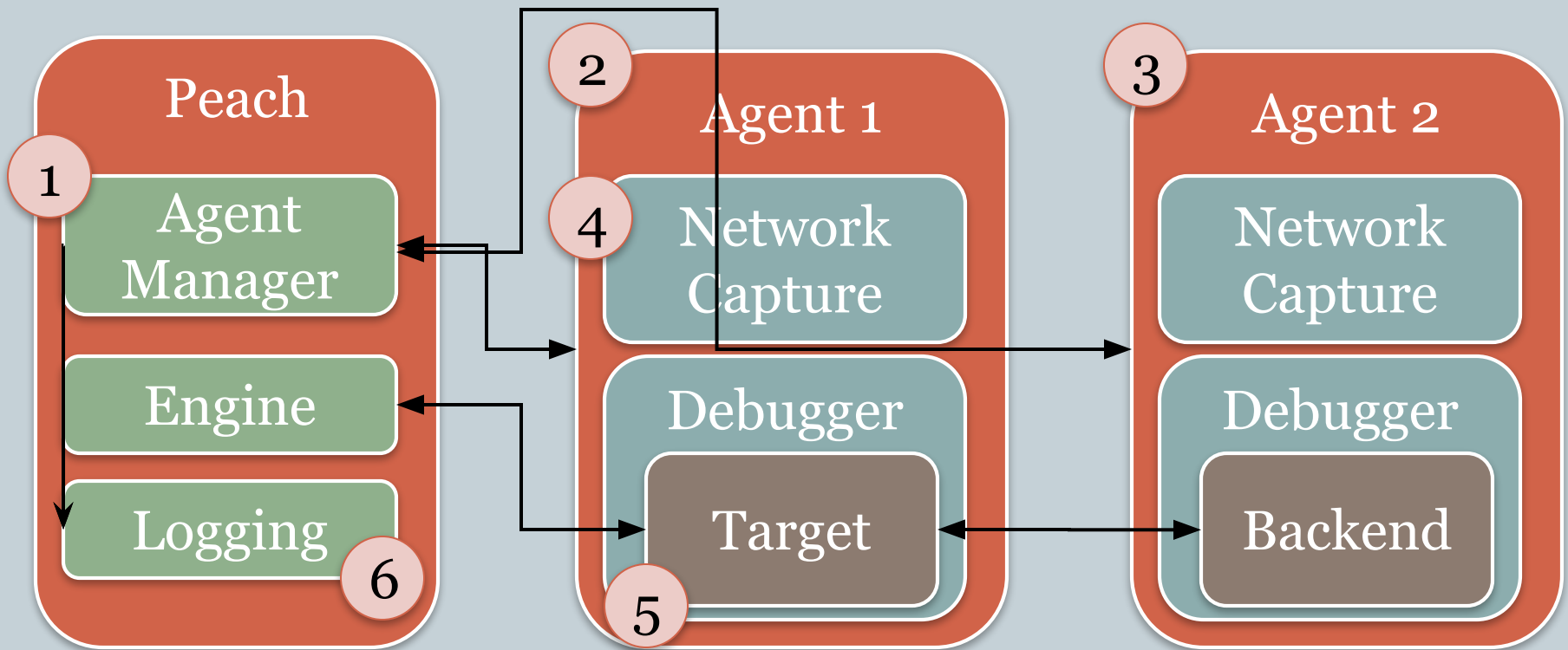


**AND DATA COLLECTION**

# Agents & Monitors



# 2 Tier Configuration



# Monitors



- Debuggers
- Process Monitor
- Memory Monitor
- Network Capture
- VM Control (snapshot, revert)
- Networked Power Strips (cycle power)
  
- Easy to implement custom monitors

# Peach Development





# Documented XML Schema



```
<Publisher class="|"></Publisher>
</Test>
<Run name="DefaultRun"
  <Test ref="HttpRequest" />
  <Logger class="logger"
    <Param name="file" />
  </Logger>
</Run>
```

- com.com
- file.File
- file.FilePerIteration
- process.Command
- sql.Odbc
- stdout.Stdout
- tcp.Tcp
- udp.Udp

Send tests to TCP port. Requires parameters named "host" and "port".

# Peach Builder

The screenshot displays the Peach Builder application window titled "Peach Builder - HelloWorld.xml". The interface is divided into several sections:

- Namespaces:** A list of XML namespaces including "default:file:defaults.xml", "pt:file:PeachTypes.xml", and "test1:file:test1.xml".
- Templates:** A tree structure showing a "Bar" template containing "Block1" and "Block2". "Block2" contains "No1" and "The String". "The String" is linked to "TheBlob" via a green arrow icon.
- Data, Agents, Tests, Runs:** A vertical list of icons for different test components.
- Relation Table:** A table with the following content:

Relation	
type	size
of	TheBlob
From	
- Buttons:** "Edit" and "Test" buttons are located at the bottom of the window.

# Peach Shark



# Peach Farm



**MASSIVELY PARALLEL FUZZING**

# Peach Farm



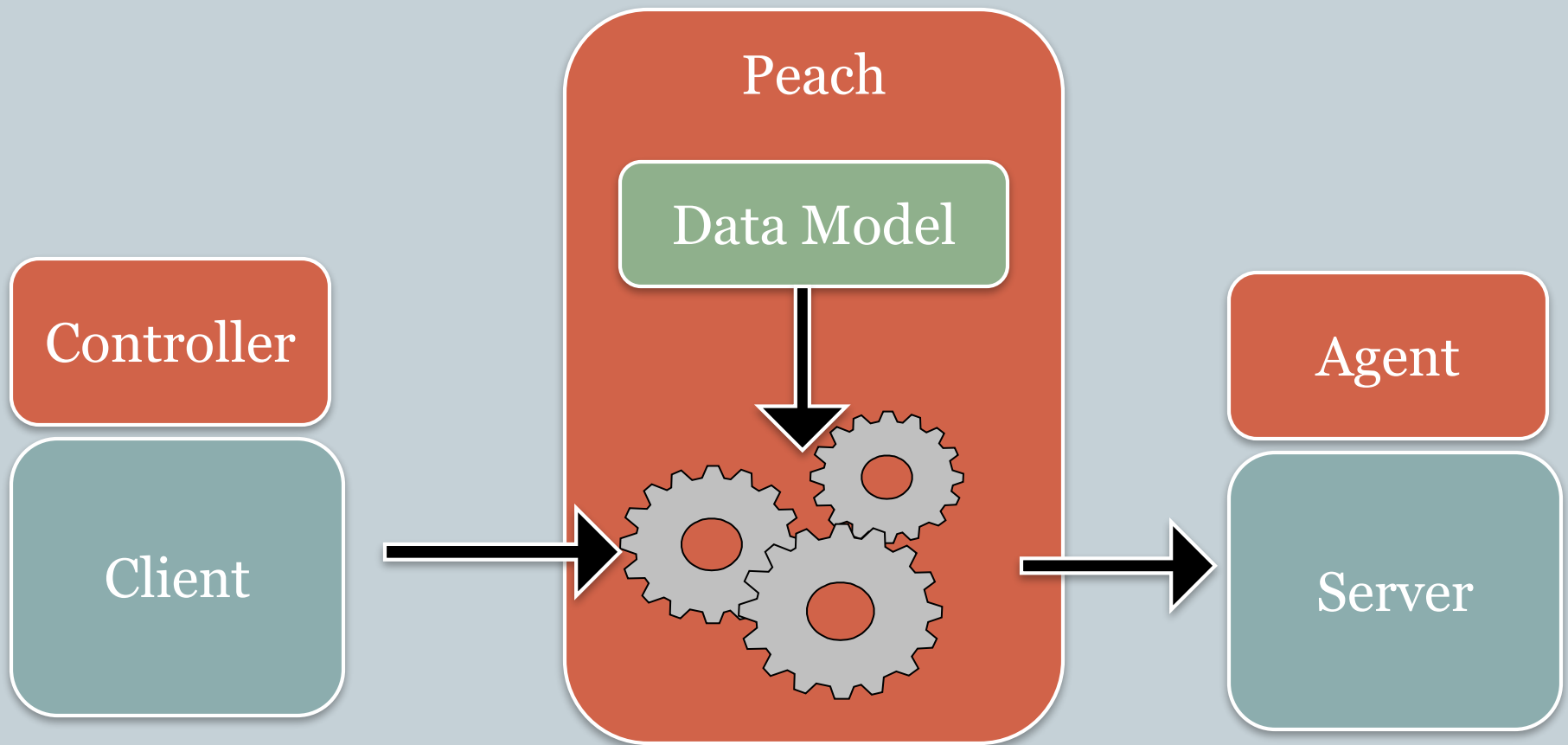
- Adam Cecchetti
- Massively Parallel Fuzzing
  - Scales from 1 to 10,000 nodes
- Choose your Virtual Platform/Hosting
  - EC2, Xen, VMWare, Etc
- Utilizes Map/Reduce Algorithm
  - Map: Maps the fuzzing cases to indexes and results
  - Reduce: Reduces fuzzing results to interesting cases
- Metric based : Time, size, diff, expected errors, OS faults, crashes

# Peach in The Middle



**WHAT'S NEXT?**

# Peach in The Middle



# Q & A



[HTTP://PEACHFUZZ.SF.NET](http://PEACHFUZZ.SF.NET)

[HTTP://PHED.ORG](http://PHED.ORG)

[MIKE@LEVIATHANSECURITY.COM](mailto:MIKE@LEVIATHANSECURITY.COM)



**leviathan**  
security group